AN EMPIRICAL POWER MODEL OF A LOW POWER MOBILE PLATFORM

A Thesis
Presented to
The Academic Faculty

by

Thejasvi Magudilu Vijayaraj

In Partial Fulfillment
of the Requirements for the Degree
Masters in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2013

AN EMPIRICAL POWER MODEL OF A LOW POWER MOBILE PLATFORM

Approved by:

Dr. Hyesoon Kim, Advisor
School of Computer Science, ECE Adjunct
Georgia Institute of Technology

Dr. Saibal Mukhopadhyay
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Sudhakar Yalamanchili
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: May 17, 2013

ACKNOWLEDGEMENTS

I would like to thank my parents, Vijayaraj and Nagamani, for all their support throughout my life. It was their constant blessings, love and encouragement that has made me stand where I am today. I take this opportunity to thank all my family members and friends for their appreciation and support.

Special thanks to my advisor, Prof. Hyesoon Kim, who gave me a chance to work on this thesis under her supervision and also for providing an opportunity to learn what research is and how things run in a PhD student's life and work. Without her guidance and persistent help, this work would not have been possible. I am also thankful for having learned some valuable lessons from her skills of managing a class and a research group.

I would like to thank all my lab members for all the support they provided, especially Sunpyo for being a mentor in the early days and for all the help he provided to identify the rights and wrongs in my ideas; Jaekyu, Nagesh and Joo Hwan for helping me setup my workstation; Pranith for walking me through the steps of solving various problems regarding the code profilers.

Finally, I would like to thank everyone who has made my life, far from home, a bit easier and enjoyable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The technology of digital design is growing so rapidly, that today's hand-held digital devices are quite comparable in performance or even better than the big old super-computers of the 90s. While these old super computers were consuming energy equivalent to what a household consumes, the current hand-held devices need to run on a small battery (of capacity around 1300 - 1500mAh) at least for half a day without recharging them. This clearly upholds the fact that power/energy is one of the today's major design constraints.

As the years pass by, consumers are expecting the hand-held devices to provide more performance, more features and simultaneously more battery life. This calls for a desperate need for power efficiency in all design corners, including the hardware and software design - hardware including general purpose processors, graphics processors, other special purpose processors, memory systems, communication systems etc and software including network data oriented applications, memory bound applications, CPU bound applications, graphics bound applications etc. Thus the need to understand the statistics, distribution and scope for decreasing the energy consumption of the system from a hardware and software perspective is high.

Power models satisfy this requirement to a certain extent, by estimating the power consumption for a subset of applications, or by providing a detailed power consumption distribution of a system. Till date, many power models have been proposed for the desktop and mobile processors/systems. These models can be broadly divided into three design levels - physical design level, circuit/RTL level and system level.

The physical design level power models utilize the circuit, gate and layout level information of the systems and are the slowest and most accurate among the three levels of power models [28, 36]. Then comes the circuit/RTL level power models which require circuit and gate level information of the systems [16, 29] and finally, the fastest of the three levels are the system level power models which require just the system level architectural information [18, 20, 21, 24, 26, 30, 31, 33, 37].

As it is almost impossible to obtain accurate layout or circuit level information on proprietary processors/systems, this work will focus on the system level power models. The process of creation of most of these system level power models involves measuring the total power consumed by the system, when different microbenchmarks stressing different blocks of the system are run on them. This work provides a method for performing accurate power measurements of individual subsystems of a mobile platform and uses this method to create an empirical power/energy model for the OMAP 4460 [9] based Pandaboard ES [10]. Also it concentrates more on energy consumption rather than power consumption, as it is total energy consumption that determines the battery life of a mobile platform.

1.1 Thesis Statement

An empirical model that predicts the energy/power consumption and distribution among several subsystems, for a given software on Pandaboard ES platform.

1.2 Organization

This document is organized as follows: Chapter II presents a new experimental setup to perform high accuracy power measurements on any platform. Chapter III explains the steps of creating an empirical power model for Pandaboard ES platform, using the experimental setup introduced in Chapter II and produces the results of energy prediction of the created model for several custom benchmarks. Chapter IV presents

the related work to the empirical power models of low power mobile platforms, different types of power models and different techniques proposed for power measurements. Chapter V concludes the document and also provides some insight into future research opportunities based on this work.

CHAPTER II

A GENERIC METHOD FOR MEASURING POWER CONSUMPTION OF AN INDIVIDUAL SUBSYSTEM IN A GIVEN SYSTEM

The easiest method of power measurement that can be used in the process of creation of an empirical model is to measure the total power the system is consuming, by inserting a power meter in between the wall socket and the device under experiment. The power consumption of the system is noted when different subsystems are stressed and this data is then used to create the power model using techniques of regression curve fitting, where the curve variables will be power consumption of different subsystems. However, the effectiveness of such a power model to predict correct power distribution among the subsystems depend majorly on the type of regression used to create it.

To eliminate the dependency of the accuracy of the model on the type of regression used in creating it, we propose measuring the power consumption of individual subsystems, when we can. This way the empirical power weights for these subsystems can be calculated directly instead of using any regression analysis. We found two methods which can be used to accurately measure the power consumption of individual subsystems. They include using the current probes and the current sense resistors.

2.1 Current Probes

Current probe is a type of probe that can be used to directly measure the current passing through a conductor. By using such current probes on the power lines of the

subsystem, one can calculate the power consumption of that subsystem. Following are the two types of commercially available current probes.

2.1.1 Split-Core Current Probes

These probes require the current carrying conductor to pass through them and are capable of measuring currents in the range of 1mA to 100s of amps. Figure 1 shows the Tektronix TCP202 split core current probe [11] which can detect current in the range of 10mA. In order to use such probes for our purpose, the power lines on the PCB, connecting to each subsystem must be disconnected and these should be connected by external wires that can pass through this current probe.

Figure 1: Tektronix TCP202 split core current probe

2.1.2 PCB Trace Current Probes

These probes work on a similar concept as of the split-core current probes, but uses a Hall effect sensor [6] at its tip, which makes it possible to directly measure the current passing through the PCB traces. The Aim I-prober 520 [2] shown in Figure 2 claims

to be the only commercially available PCB trace current probe. By simply locating and placing this probe on a power line of a subsystem, one can measure the current passing through it. However, there are some critical issues that needs to be taken care of while using these probes. These include the fact that the current through the neighboring PCB traces affect the current measurement and also the fact that these probes should be calibrated before each use, as the orientation of the probe with respect to the earth's magnetic field will affect the measurements slightly.

Figure 2: Aim I-prober 520 PCB trace current probe

2.2 Current Sense Resistor

This idea includes inserting or using the already existing components like current sense resistors or small inductors in series of the power lines of the subsystems of

the device and then measure the voltage drop across these sense resistors. With the accurate value of the resistance of the sense resistor and the power supply voltage to the corresponding subsystem, the power consumption can be calculated.

It is recommended, not to rely on huge inductors as sense resistors, on power lines connected to subsystems that switch a lot of transistors frequently, like inductors on the power lines for the processors. Such subsystems consume current in form of spikes when a large number of transistors are switching and these current spikes will cause either positive or negative induced voltage across the inductor and will affect our energy measurement. Figure 3 shows the voltage drop across the 1uH inductor on the power supply line of the Cortex-A9 cores on Pandaboard ES and Figure 4 shows the voltage drop across a sense resistor placed in series of this inductor, for the same current passing through both of them. It can be noted that there is a significant amount of induced voltage across the inductor and it also goes negative when the current through the inductor reduces. The positive induced voltage will cause the measured energy to be more than the actual energy consumption and the negative induced voltage will give a lower estimate of energy consumption.

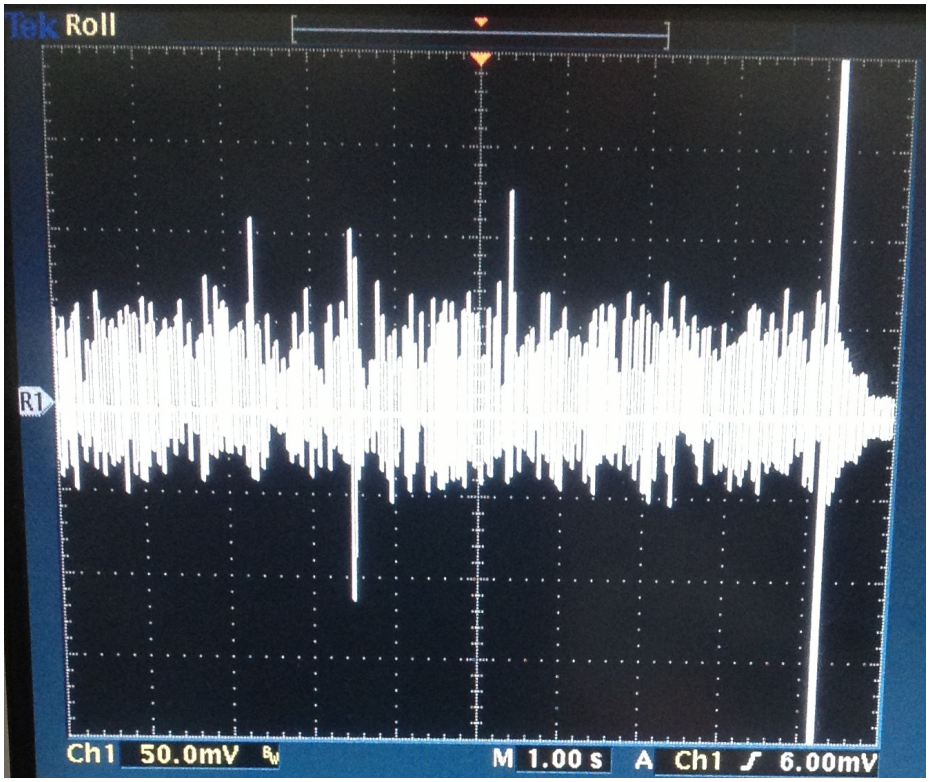

Figure 3: Resistive drop and induced voltage across the series inductor on the power line of Cortex-A9 cores

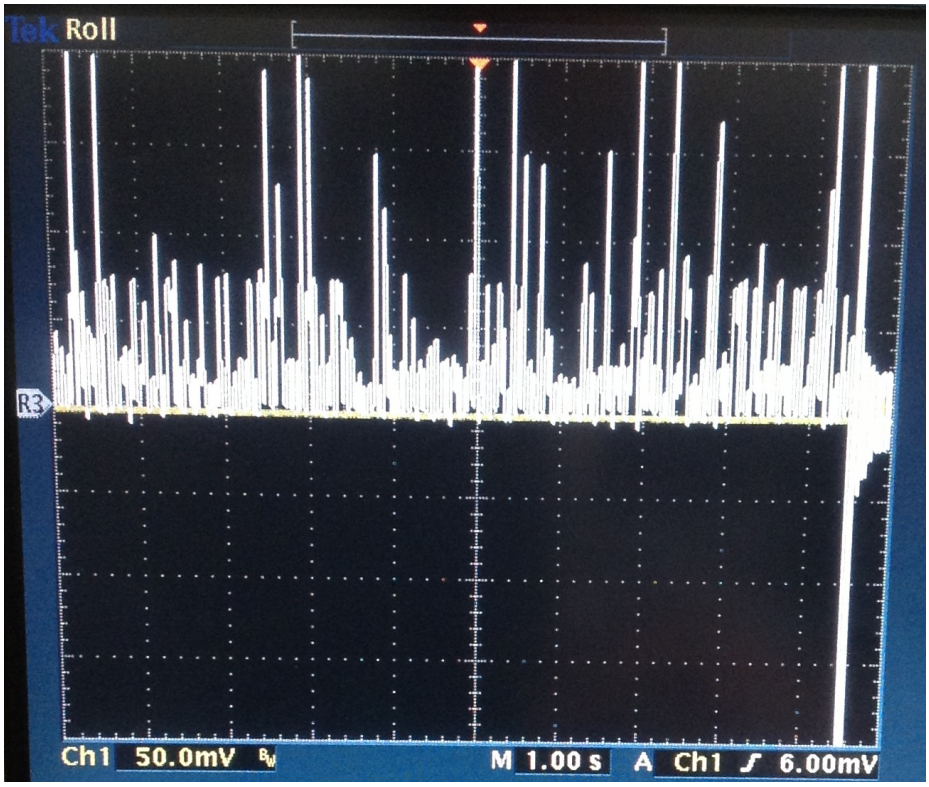

Figure 4: Resistive voltage drop across the sense resistor on the power line of Cortex-A9 cores

2.2.1 Choice of Current Sense Resistor

The current sense resistors must be chosen such that, the drop across them will be higher than the accuracy of the multimeter or an oscilloscope that is being used. Equation (1) provides a relation to find out the minimum resistance of the sense resistor, so that the voltage drop across it will be read accurately by the multimeter or oscilloscope being used.

$$\text{Value of current sense resistor} > \frac{\text{Accuracy of the voltage measuring device}}{\text{Minimum current passing through the power line}} \quad (1)$$

Also the value of the sense resistor is limited by the maximum amount of voltage drop allowed on the power lines. Equation (2) provides a relation to obtain the maximum resistance value that can be used as a sense resistor on the power line of a

particular subsystem.

$$\text{Value of current sense resistor} < \frac{\text{Supply voltage} - \text{Minimum supply voltage required by the subsystem}}{\text{Maximum current passing through the power line}} \quad (2)$$

However, it is not necessary to calculate the value of each sense resistor using the above equations, but these equations can be used for debugging, when the system stops working or is not acting normal when the sense resistor is inserted.

2.3 The Chosen One

We chose to use the current sense resistors for the power measurement over using the current probes, as this method requires less calibration and also provides higher accuracy of measurement. The following sections explain the experiment setup and steps of obtaining accurate value of power consumption by different subsystems.

2.3.1 Probe Station

The probe station [7] is a sturdy setup to latch the printed circuit board under experiment, while probes are attached to it for power measurements. It consists of the following elements from Cascade Microtech.

- Module Test Base Station (MTBS) - is a heavy base plate which serves as a stable immovable platform for the experiment setup.
- Board mounting station - is a station where the board being used for the experiment is latched onto.

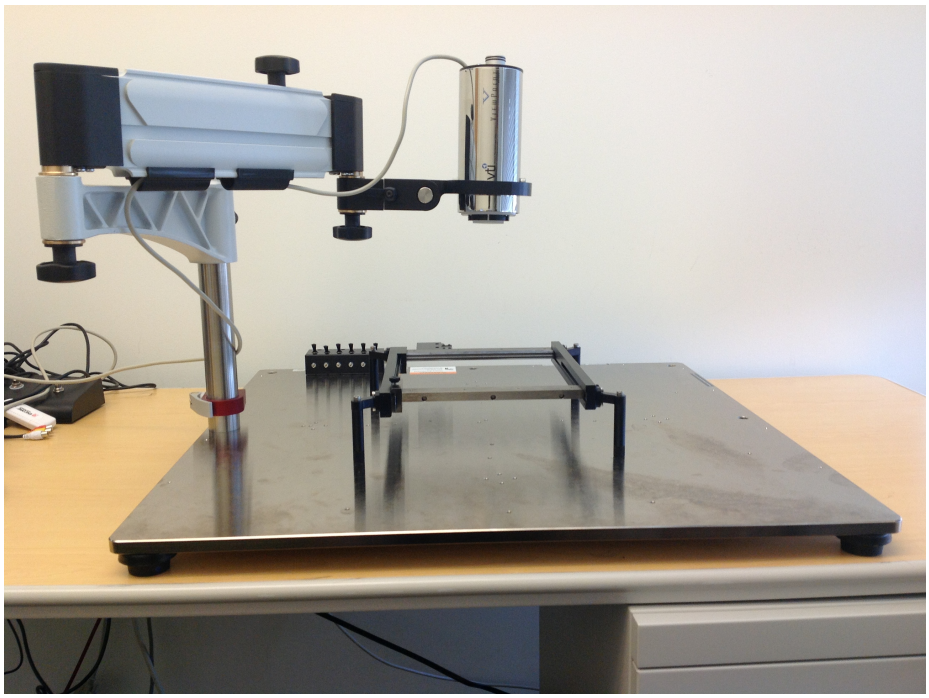

Figure 5: MTBS with the board mounting station and the microscope

- EZ-Probe positioner [5] - is a precise positioner which provides fine movement and placement of the probe in X-Y axis. This positioner is also equipped with a base that is capable of holding strongly onto the MTBS on providing a suction pressure.

Figure 6: EZ-Probe positioner

- Fine-pitch DC needle holder - is used as an accessory to the EZ-Probe positioner to hold the tungsten needle in place.

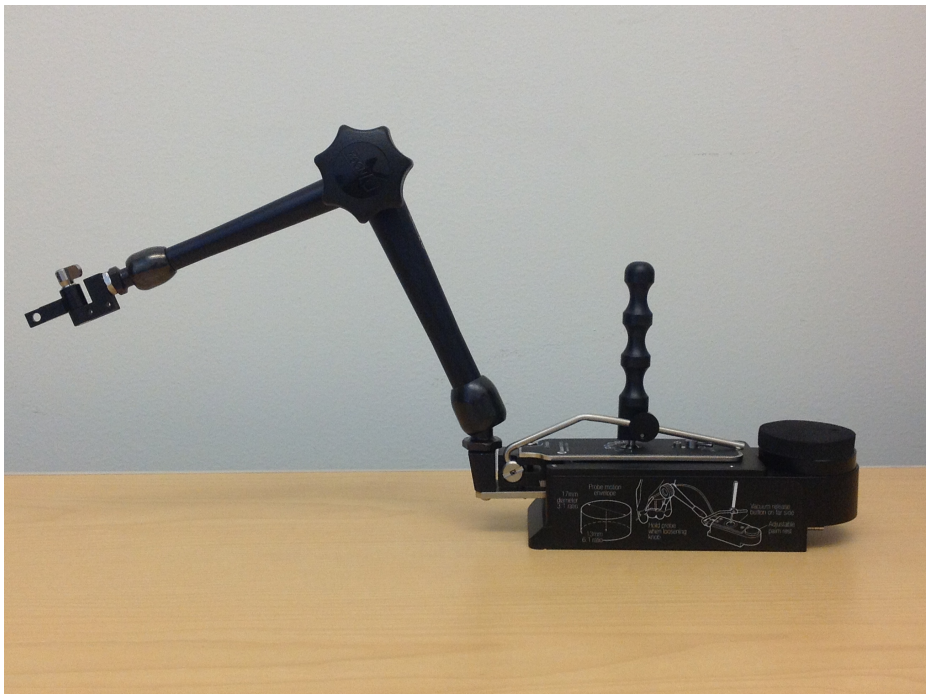

Figure 7: EZ-Probe positioner with the fine pitch DC needle holder

- Probes - The probes used are low resistance tungsten tip needles with a tip radius of 5um.

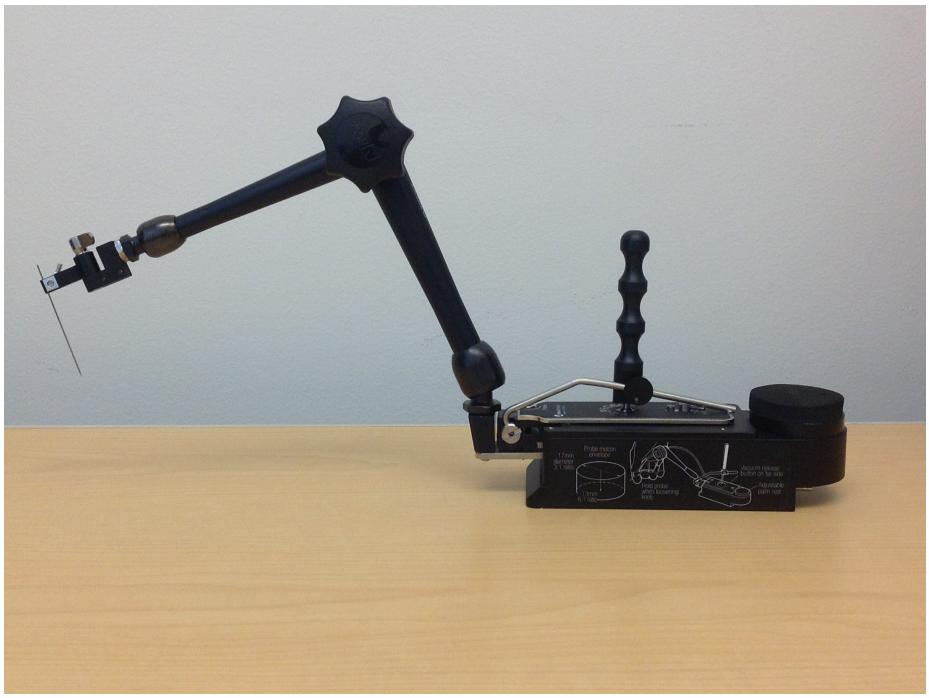

Figure 8: Fine pitch tungsten tip probe attached to the EZ-Probe positioner

- Vacuum pump - is used to provide the suction pressure to the EZ-Probe positioners.

2.3.1.1 Probe Setup

The probes used to measure the voltage drop across the current sense resistors are setup by following the below steps.

- Attach the fine-pitch DC needle holder to the EZ-probe handler.
- Insert the fine-pitch tungsten tip needle to the DC needle holder.
- Attach the multimeter probe to the probe setup.

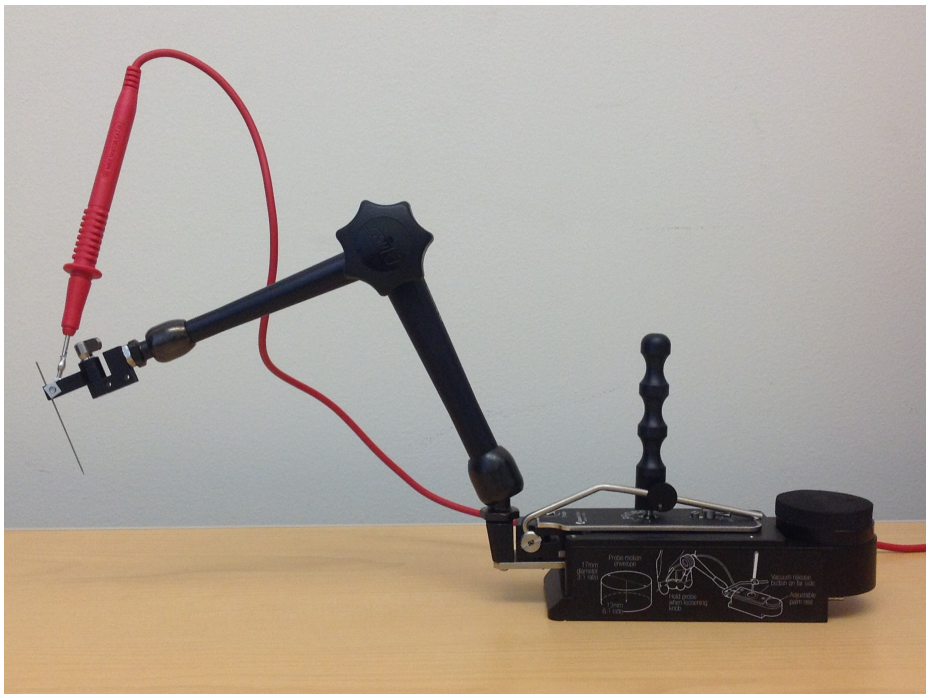

Figure 9: Complete probe setup

2.3.1.2 Probe Station Setup

The probe station which involves all components to perform the power experiments is setup by following the below steps.

- Attach the board under experiment to the board mounting station.

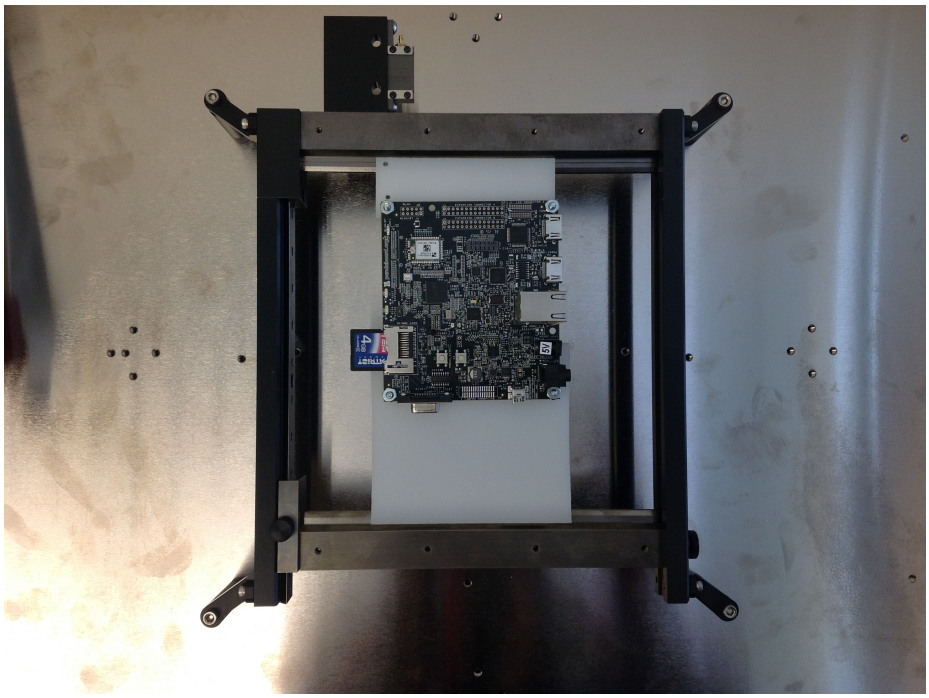

Figure 10: Board attached to the board mounting station

- Place the probes at the pads of the sense resistor across which the voltage drop should be measured, using the x-y axis precise motion joystick of the EZ-probe handler.

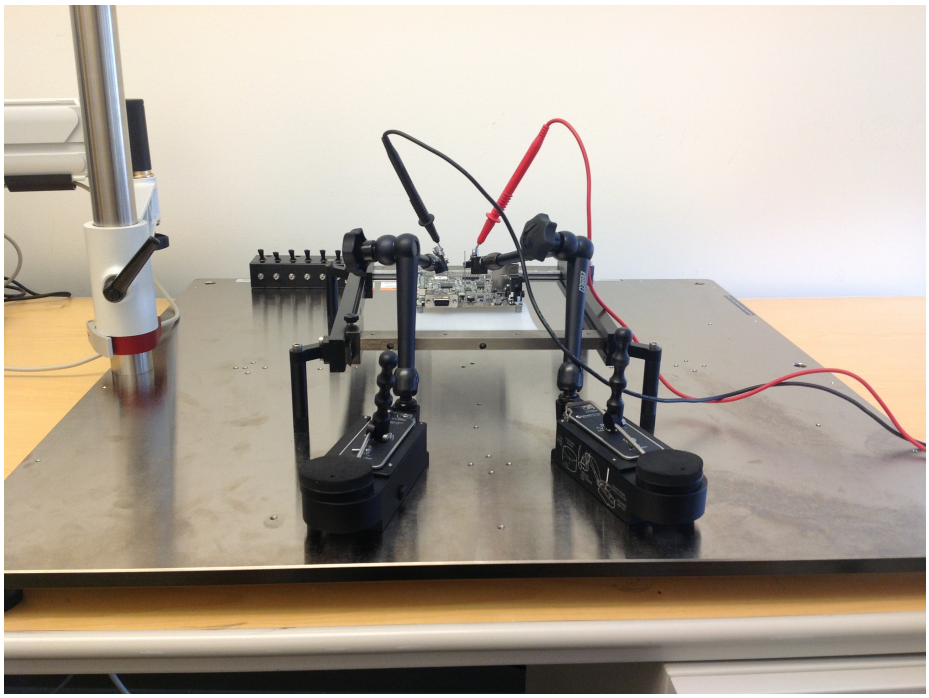

Figure 11: Probes placed on the points of interest on the board

- Turn on the vacuum pump, so that the probes stay in place even if the entire setup is disturbed.
- Connect the multimeter or oscilloscope probes to the measuring device.

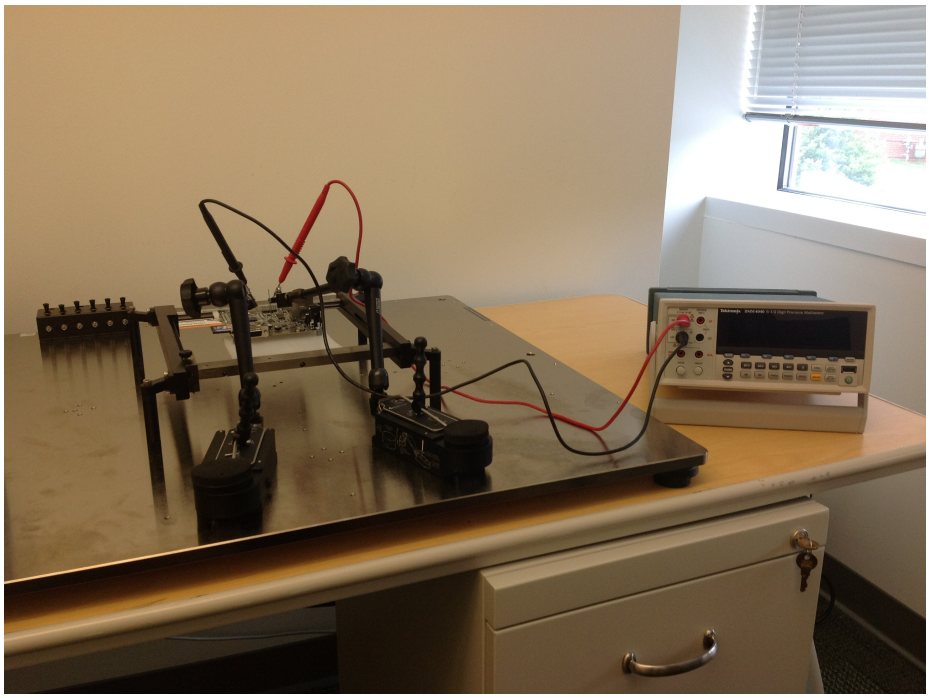

Figure 12: Complete probe station setup

2.3.2 Steps of Measuring Power Consumption of a Subsystem

Once the probe station and the board under experiment are setup, one can follow the steps below to obtain accurate power measurement of a subsystem.

Measure the probe resistance - Short the probes at their tips and measure the total probe resistance. This must be done, so that the accurate value of the resistance of the sense resistor can be calculated.

Using the probes measure the resistance of the sense resistor, then subtract the probe resistance from the measured value to get the actual sense resistor value.

$$\text{Actual resistance} = \text{Measured resistance of the resistor} - \text{Probe resistance} \quad (3)$$

Measure the supply voltage to the subsystem under test. This value will be used to measure the total power consumption. In case the supply voltage is dynamically scaled, then a pair of probes must be setup to measure the supply voltage continuously

when the power consumption of the subsystem is being measured.

Measure the voltage drop across the resistor during the experiment. The current that the subsystem consumed can be calculated from the measured voltage drop by the using the Equation (4).

$$\text{Current through the resistor} = \frac{\text{Voltage drop across the resistor}}{\text{Actual resistance}} \tag{4}$$

While calculating the power consumption, the voltage drop across the resistor should also be considered and the power dissipated in this resistor should be subtracted from the total power calculated. Let E_m denote the energy consumed by a module, I be the current consumed by the module, V_{CC} be the power line voltage of the module, V_R be the voltage drop across the sense resistor and τ_s be the sampling period of the measuring device, then - assuming that the current consumption as constant within one sample period - the relation between these entities is provided by Equation (5).

$$E_m = I \times (V_{CC} - V_R) \times \tau_s \tag{5}$$

$$\text{Total power consumption} = \frac{\sum \text{Energy consumption}}{\text{Total time}} \tag{6}$$

CHAPTER III

AN EMPIRICAL POWER MODEL FOR OMAP 4460 BASED PANDABOARD ES

The Pandaboard ES based on the OMAP 4460 SOC was chosen for creating an empirical power model using the method discussed in the previous chapter. The major reason for choosing Pandaboard is that it is open source and has huge support from Texas Instruments, the Android and Linux communities.

3.1 Pandaboard ES Architecture

The Pandaboard ES [10] consists of the OMAP 4460 SOC [9] with a dual-core Cortex-A9 processor [4] as its backbone. It has an 8GB Package-On-Package LPDDR2 memory, TiWi-R2 BLE Wi-Fi module [13], TPS62631 switching power supply powering the ARM cores, TWL6030 power management IC powering other components on the board, LAN9514 USB hub and Ethernet controller, JTAG, UART, DVI-D and HDMI connectors, audio jacks and SD / MMC card cage. Figure 13 shows the block diagram of the Pandaboard ES architecture.

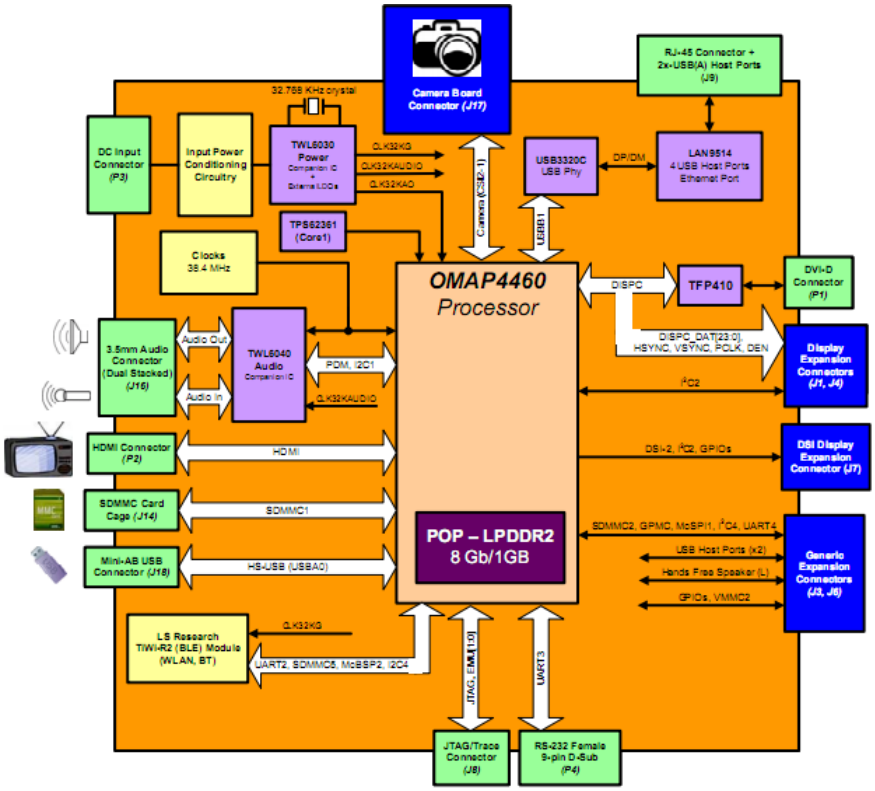

Figure 13: Pandaboard ES architecture

3.1.1 OMAP 4460 Architecture

OMAP 4460 high-performance multimedia device consists of a dual-core Cortex-A9 cores with 32KB L1 instruction and data caches and 1MB L2 cache, digital signal processor, image and video accelerators, Cortex-M3 subsystem including two ARM Cortex-M3 microprocessors, audio back-end subsystem, image signal processor, still image coprocessor, display subsystem, SGX 540 2D/3D graphics accelerator. This device is capable of streaming video up to full HD, 2D/3D mobile gaming, video conferencing and high resolution still imaging. This device supports variety of operating systems including Linux, Android, Symbian OS, Windows CE, WinMobile. Figure 14 shows the block diagram of the OMAP4460 architecture.

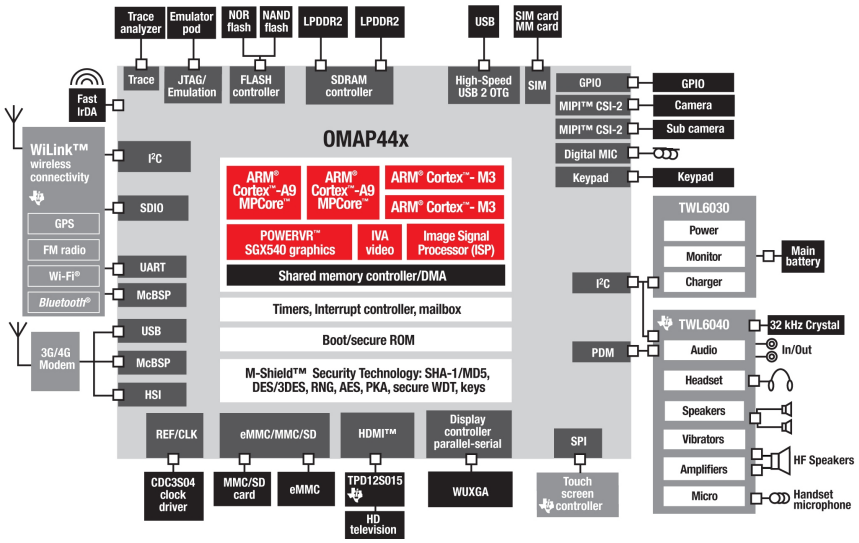

Figure 14: OMAP 4460 architecture

3.2 Preparation of the Board for Experiment

3.2.1 Hardware preparation

Hardware preparation of the board for experiment involves inserting the current sense resistors in series of the power lines of different subsystems. For the experiments, we inserted current sense resistors of the range from 0.02 ohms to 0.5 ohms in series with the inductors on the power lines. Figure 15 indicates a position for inserting the sense resistor for the MPU (Micro-Processor Unit) subsystem power line of the Pandaboard ES.

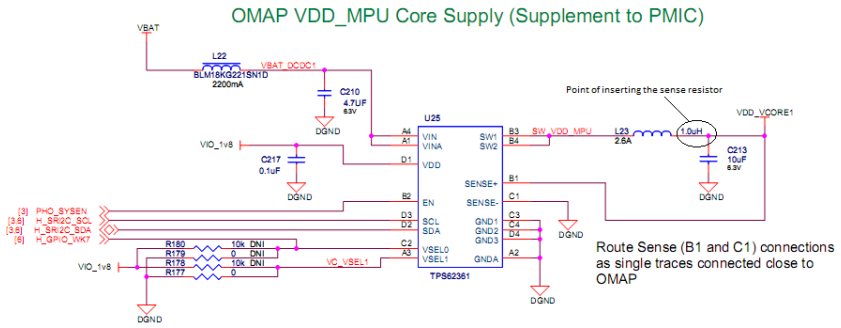

Figure 15: Point for inserting the sense resistor for the MPU subsystem power line

The sense resistors were inserted in series with the inductor L23 - placed on the line powering the MPU subsystem consisting of the Cortex-A9 cores, inductor L9 - placed on the line powering the LPDDR2 memory, inductor L14 - placed on the line powering the CORE subsystem consisting of the Cortex-M3 cores, SGX 540 graphics accelerator, image subsystem and face detection system, inductor L16 - placed on the line powering the audio, video and DSP subsystem. Table 1 provides a detailed list of power/voltage domains of Pandaboard ES and their corresponding power-line inductors and the values of sense resistors inserted in series with the inductors.

Table 1: Power/voltage domains of Pandaboard ES

Power/voltage Domain	Subsystems	Power-line inductor	Value of the inserted sense resistor (ohm)
VDD_MPU	Two Cortex-A9 CPUs with 32-KB L1 I/D cache, L2 cache and PL310 L2 cache controller	L23	0.05
VDD_CORE	SGX540 graphics accelerator, two Cortex-M3 microcontrollers and imaging subsystem	L14	0.05
VDD_LPDDR	1GB POP LPDDR2 DRAM	L9	0.2
VDD_IVA	Audio, video and DSP subsystem	L16	0.5
VDD_REST	Wi-Fi, SD-card and rest of the subsystems	L21	0.02

3.2.2 Software preparation

Software preparation includes onetime building or deploying of an operating system of choice on the Pandaboard ES. We built the Android 4.1.2 with Pandaboard binaries and modified kernel 3.2. This step also includes writing stressing microbenchmarks

for different subsystems and validating them.

3.3 Subsystem Stressing Microbenchmarks

Several microbenchmarks were written to stress different subsystems of the Pandaboard ES while their power consumption was being measured. These can be broadly classified into major categories as computational, memory, Wi-Fi and GPU microbenchmarks. All these benchmarks were combined as one Android app [3].

Figure 16: Screenshot of the benchmark suite used for the experiments

3.3.1 Computational Benchmarks

These benchmarks are targeted towards stressing the computational units of the Cortex-A9 subsystem, such as the integer and floating point ALUs, multipliers and dividers.

3.3.2 Memory Benchmarks

Memory benchmarks include the L1, L2 and DDR benchmarks. The L1 benchmark repeatedly accesses data from an array that fits in the L1 cache. The L2 benchmark creates an array of size much greater than the L1 cache capacity (32KB) and continuously accesses data from different cache lines, which are not cached in the L1 cache.

The DDR benchmark follows the lines of the L2 benchmark by creating an array of size much greater than the L2 cache capacity (1MB). All of these benchmarks were validated using OProfile on Android.

3.3.3 Wi-Fi Benchmarks

Wi-Fi benchmarks include the transmit and receive benchmarks. The receiver benchmark continuously downloads a huge file (512MB) but discards the data, as we do not want to stress the DDR or the SD card while running this benchmark. Whereas the transmit benchmark uses the NanoHTTPD server [8] and serves a huge file when a HTTP request is made from a browser.

The benchmark app on Android locks the Wi-Fi and network state as soon as it becomes active, so that no other app or the OS can modify the Wi-Fi state while the benchmarks are running. The transmit power and the power management schemes of the Wi-Fi controller were modified using the $iwconfig$ Linux command, to measure the power consumption of the Wi-Fi module under different conditions. However, the benchmark app should be paused while changing the Wi-Fi state using $iwconfig$, as the app holds the Wi-Fi state lock when it is active, and should be resumed only after the necessary changes are completed.

3.3.4 GPU Benchmarks

The graphics benchmarks of the 0xbench [1] application for Android were used to stress the SGX 540 subsystem of the OMAP 4460.

3.4 Experimental Results

All the power experiments were performed on the Pandaboard ES modified suitably as described in the board preparation section. The measurements during the experiments were taken using the Tektronix DMM4040 6-1/2 digit precision multimeter [12] with a full 6-1/2 digit resolution at a sampling rate of 10 PLC (Power Line Cycles).

All the measurements taken by the multimeter were stored on a PC connected to it, using the inbuilt data-logger facilities. Figure 17 provides an abstract block diagram of the experimental setup.

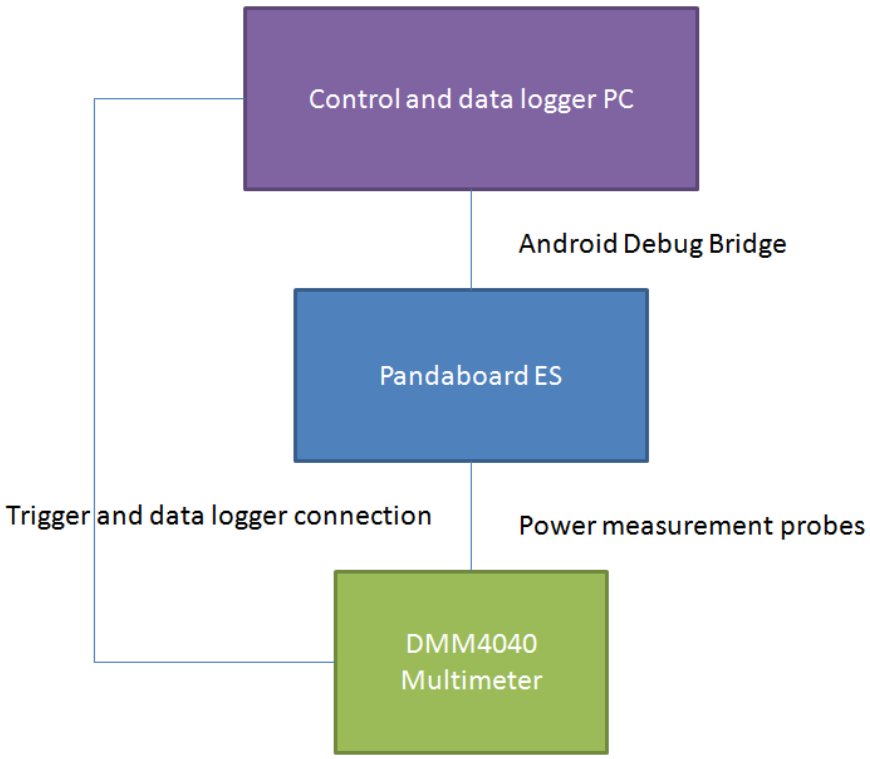

Figure 17: Block diagram of the complete experiment setup

Several experiments were performed by running all the benchmarks and the energy consumption of different subsystems were recorded. All the energy consumption values produced below correspond to the sampling rate of 10 PLC, which is (10 * (1/60Hz)), i.e 0.1667s. Table 2 lists all the benchmarks/cases run for obtaining the energy consumption values of different subsystems. Table 3 provides the total energy consumption of major subsystems when the computational and memory benchmarks were executed for 60 seconds.

Table 2: Description of subsystem stressing benchmark cases

Benchmark abbreviation	Name of the benchmark	Description
JA	Just Android	In this case, the system will be just running Android OS and no other application will be running
Idle	Idle or sleep mode	In this case, the system will be put to sleep.In this mode, all the subsystems are put into deep sleep/power saving states
IA	Integer Addition	This benchmark continuously executes integer addition instructions on different data sets
IM	Integer Multiplication	This benchmark continuously executes integer multiplication instructions on different data sets
ID	Integer Division	This benchmark continuously executes integer division instructions on different data sets
FA	Floating point Addition	This benchmark continuously executes a mixture of single precision and double precision floating-point addition instructions on different data sets
FM	Floating point Multiplication	This benchmark continuously executes a mixture of single precision and double precision floating-point multiplication instructions on different data sets
FD	Floating point Division	This benchmark continuously executes a mixture of single precision and double precision floating-point division instructions on different data sets
L1	L1 cache	This benchmark continuously accesses memory locations that are cached in the L1 cache
L2	L2 cache	This benchmark continuously accesses memory locations that are not cached in the L1 cache but in the L2 cache
DDR	LPDDR2 DRAM	This benchmark continuously accesses memory locations that are not at all cached and are in the LPDDR2 DRAM
WiFi-DLD-PM-ON	Wi-Fi download with power management on	This benchmark continuously downloads a 512MB file but discards the data. The power management of the Wi-Fi is turned on for this benchmark case
WiFi-DLD-PM-OFF	Wi-Fi download with power management off	This benchmark continuously downloads a 512MB file but discards the data. The power management of the Wi-Fi is turned off for this benchmark case
WiFi-server-20dbm-PM-ON	Wi-Fi server with transmit power of 20dbm and power management on	This benchmark serves a huge file on request with the Wi-Fi transmit power set to 20dbm and with the power management on
WiFi-server-20dbm-PM-OFF	Wi-Fi server with transmit power of 20dbm and power management off	This benchmark serves a huge file on request with the Wi-Fi transmit power set to 20dbm and with the power management off
WiFi-server-15dbm-PM-ON	Wi-Fi server with transmit power of 15dbm and power management on	This benchmark serves a huge file on request with the Wi-Fi transmit power set to 15dbm and with the power management on
WiFi-server-15dbm-PM-OFF	Wi-Fi server with transmit power of 15dbm and power management off	This benchmark serves a huge file on request with the Wi-Fi transmit power set to 15dbm and with the power management off
WiFi-server-10dbm-PM-ON	Wi-Fi server with transmit power of 10dbm and power management on	This benchmark serves a huge file on request with the Wi-Fi transmit power set to 10dbm and with the power management on
WiFi-server-10dbm-PM-OFF	Wi-Fi server with transmit power of 10dbm and power management off	This benchmark serves a huge file on request with the Wi-Fi transmit power set to 10dbm and with the power management off
WiFi-server-5dbm-PM-ON	Wi-Fi server with transmit power of 5dbm and power management on	This benchmark serves a huge file on request with the Wi-Fi transmit power set to 5dbm and with the power management on
WiFi-server-5dbm-PM-OFF	Wi-Fi server with transmit power of 5dbm and power management off	This benchmark serves a huge file on request with the Wi-Fi transmit power set to 5dbm and with the power management off
WiFi-server-0dbm-PM-ON	Wi-Fi server with transmit power of 0dbm and power management on	This benchmark serves a huge file on request with the Wi-Fi transmit power set to 0dbm and with the power management on
WiFi-server-0dbm-PM-OFF	Wi-Fi server with transmit power of 0dbm and power management off	This benchmark serves a huge file on request with the Wi-Fi transmit power set to 0dbm and with the power management off
2D	2D graphics benchamark	This is the 2D canvas benchmark of the 0xbench Android benchmark suite
3D	3D graphics benchamark	This is the 3D canvas benchmark of the 0xbench Android benchmark suite

Table 3: Total energy consumption (J) of major subsystems for variety of computational and memory microbenchmarks

Subsystem	Just Android	Sleep	IntAdd	IntMul	IntDiv	FloatAdd	FloatMul	FloatDiv	L1	L2	LPDDR2 DRAM
MPU	3.516	3.499	46.542	38.559	58.108	41.439	40.972	35.754	62.301	53.073	46.788
CORE	24.981	24.976	23.626	23.706	24.709	24.984	24.979	24.979	24.980	24.981	29.263
LPDDR2	9.562	5.546	9.559	9.562	9.561	9.562	9.559	9.561	9.558	9.560	15.352
Rest	86.476	58.317	96.709	96.405	103.755	99.868	96.531	99.503	100.205	103.159	83.891

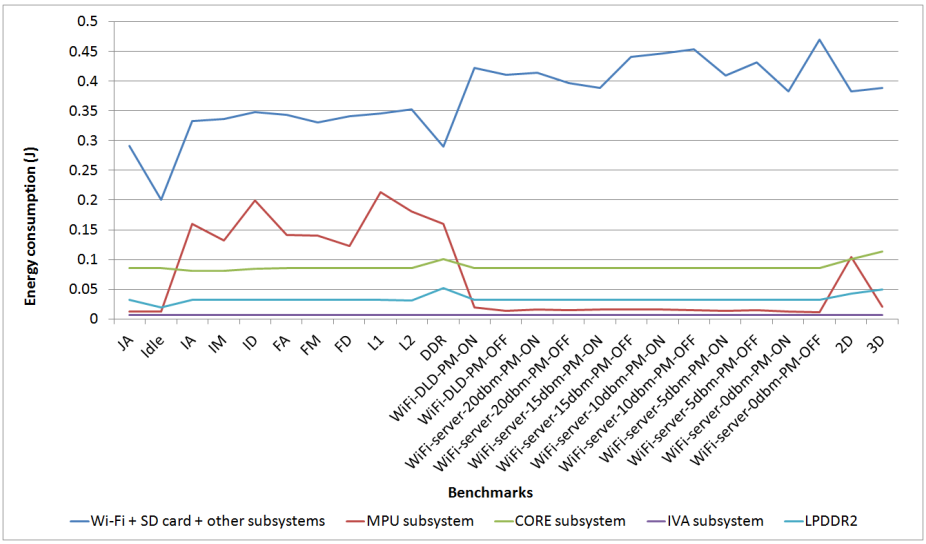

Figure 18: Average energy consumption of different subsystems on running variety of microbenchmarks

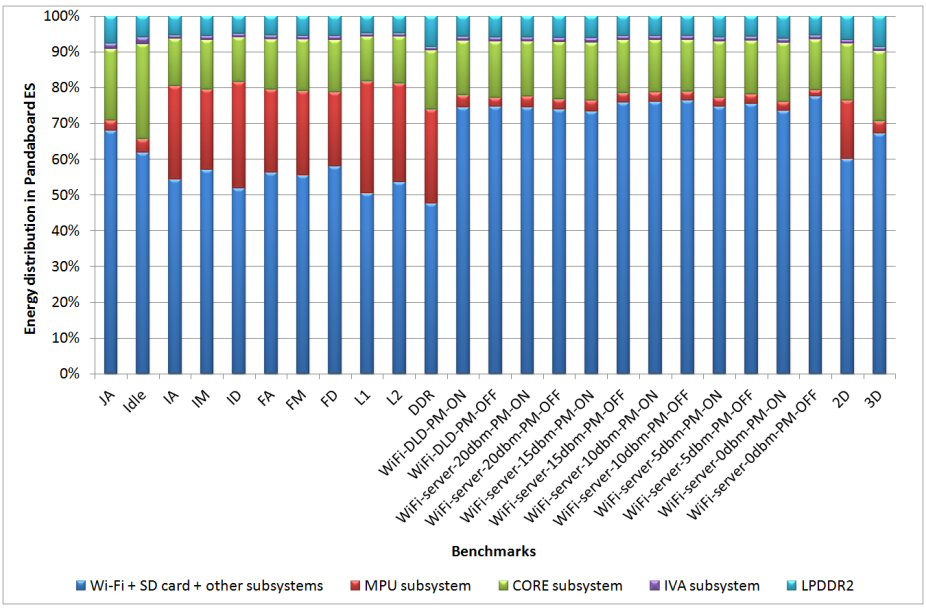

Figure 19: Energy distribution in Pandaboard ES on running variety of microbenchmarks

Figure 18 shows the average energy consumption by different subsystems of the Pandaboard ES when variety of stress benchmarks are run on it. Figure 19 indicates the energy distribution across the entire Pandaboard ES under different conditions. The average energy specified here are all for a sample period of 10PLC or 0.1667s. All benchmarks except the Wi-Fi server benchmarks were executed for a duration of 60 seconds.

It can be noted that the MPU subsystem, which includes the Cortex-A9 cores, consumes just around 0.012 J of average energy when idle or put to sleep. The reason for such a low energy consumption during idle states, is the presence of power and clock gating options in the OMAP 4460. The average energy consumption of the MPU subsystem stays almost same as the idle average energy consumption when variety of Wi-Fi benchmarks are running, as for these benchmarks, the utilization factor of the CPU is very low. However, the increase in the average energy consumption can be noted for the computational and graphic benchmarks. The computational

benchmarks show around an average of 0.14 J up to a maximum of 0.2 J of increase in the energy consumption of the Cortex-A9 cores. From the above figure it looks like the average energy consumption for the DDR benchmark is lesser than the L2 benchmark which is lesser than the L1 benchmark. But to obtain the total energy consumed by a request to the L2 cache or the DDR2 memory, one should consider the latency of the request. As the latency of the L2 cache is in the order of 8-10 cycles in OMAP 4460 and DDR2 access latency being 100s of processor cycles, the total energy, which is higher than the energy consumed if the data was in L1 cache, is spread over this latency period and appears to be less.

The LPDDR2 DRAM consumes an average of 0.032 J of energy under minimal utilization. When the entire system is in idle/sleep mode, the average energy consumption reduces to around 0.019 J. The highest amount of energy consumed by this subsystem is when the DDR2 stress benchmark was run, with the average energy consumption reaching its peak at 0.052 J.

The CORE subsystem, which includes the Cortex-M3 cores, SGX 540 graphics accelerator and the image and face detect systems, consumes an average of 0.085J of energy under normal workload, with the peak average energy consumption of 0.114J when the 3D graphics benchmarks are run.

The IVA subsystem, which includes the audio, video and DSP subsystems, is not stressed by any of the benchmarks in this work, as we are not working with displays and audio systems. However, the base energy consumption of this subsystem cannot be neglected while calculating the total power consumption. It can be noted that the IVA subsystem, under very minimal stress, consumes around 0.006J of energy.

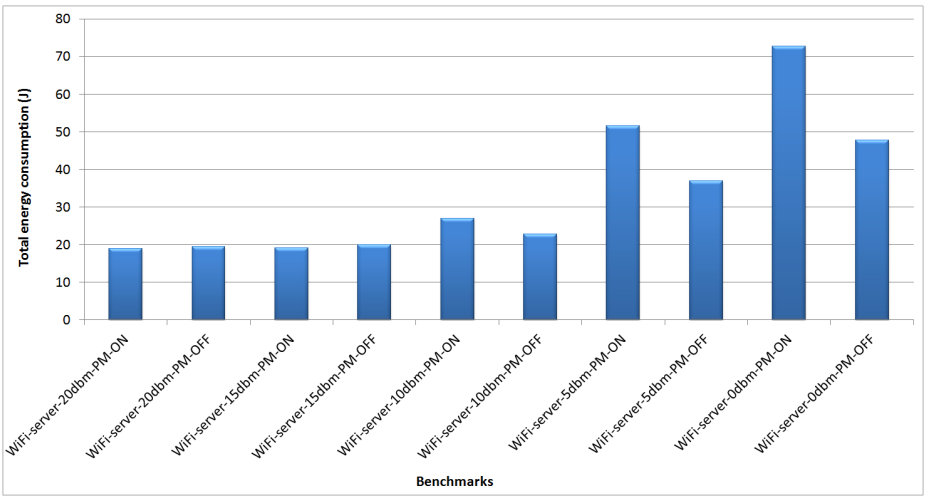

Figure 20: Total energy consumed by the Wi-Fi module under different transmit power settings

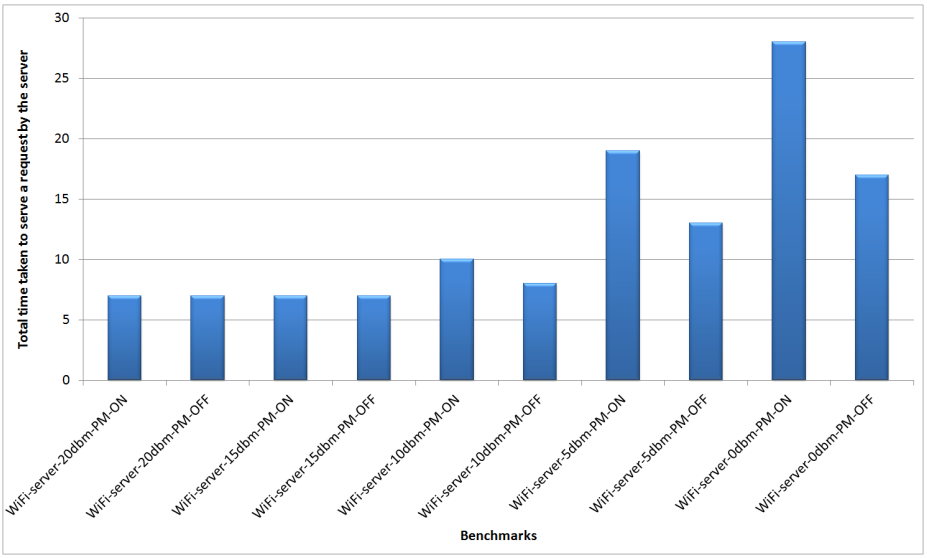

Figure 21: Total time taken to completely service a request by the server under different Wi-Fi transmit power settings

Figure 20 shows the total energy consumption of the Wi-Fi module and Figure 21 shows the total time it took for the server to complete a request, which is served by sending a 4.62MB file, under a given setting, when the Wi-Fi server benchmarks are run. The Wi-Fi server benchmarks, locks the Wi-Fi state when started, so that no

other app or even the OS can change the state of the Wi-Fi until it unlocks the state. This way the Wi-Fi module is not put to sleep by any power management features of the OS or any apps installed on Android. However, the built-in power management features of the Wi-Fi module can still be turned on or off whenever required.

The Wi-Fi server benchmarks were run by setting the transmit power of the Wi-Fi module at 20, 15, 10, 5 and 0dbm with power management turned on/off and the total energy consumption was recorded. From the figures, it can be deduced that the total amount of energy consumed is least when the Wi-Fi module has a transmit power of 20 or 15dbm and it increases rapidly by around 4 fold as the transmit power is decreased to 0dbm. The reason for this increase being, the lower bitrates and the increase in number of failed transactions at low transmit powers and thus the increase in the total time required to serve a request.

The Wi-Fi power management schemes perform better at higher bitrates or transmit power, but fail significantly when the bit-rates are too low by performing automatic Wi-Fi state changes and increasing the total time to serve a request by the server.

3.5 Creation of the Empirical Model

The empirical power/energy model was created based on the energy measurements performed on different subsystems of the Pandaboard ES while benchmarks stressing these subsystems in different ways were executed.

The steps used in the creation of the empirical model uses the following assumptions or facts .

- Energy consumed by a module/subsystem in one clock cycle remains constant even if the module clock frequency changes. The reason behind this assumption is as follows - Whenever the clock of a module makes a transition and causes some transistors in it to switch, which in-turn will cause some intermediate

nodes of the module - which were at ground potential or some other potential - to be connected the supply voltage, current flows from the power supply line to this intermediate node, so that its potential rises from its previous level to the supply voltage level. This current is major cause of energy consumption. If the modules do not switch any transistors and stay at whatever state they are in, for any amount of time, they will not consume any dynamic energy. However, there will be static energy loss through the transistors, but this work does not provide much attention on static energy consumption. Therefore, extending this logic, it means that a module should consume the same amount of energy if a node of it, which is at ground state, is connected to the supply voltage, irrespective of the clock frequency. Because it is the number of node transitions from ground or low potential to high potential, that determines the total dynamic energy consumption of the module. However, the power consumption of a module changes with frequency, as power consumption is the density of energy consumption in a unit time. Thus if the frequency increases, the total energy consumption for a given task remains the same, but the task completes at an earlier time, increasing the power consumption. Similarly, when the frequency decreases, the power consumption decreases.

- Current consumption of each subsystem is assumed to be constant over the measurement sampling period. The Tektronix DMM4040 digital multimeter was used to measure the voltage drop across the sense resistors and these measurements were done with a sampling period of 10PLC (≈ 0.1667s). The average energy of each subsystem was calculated over this sampling period, over which the utilization of the subsystem was almost kept constant by the microbenchmarks. Thus the model makes the assumption that energy distribution is even and smooth over any small period of time considered within each sampling period.

- Energy consumption of a subsystem under maximum utilization, for one clock cycle is calculated by Equation 7, where $E_{avg_{\tau_s}}$ is the average energy consumption of the module under maximum utilization over a measurement sampling period, τ_s is the measurement sampling period and τ_{clk} is the clock period for this module while the energy measurements were taken.

$$E_{module_{perClk}} = \frac{E_{avg_{\tau_s}}}{\tau_s} \times \tau_{clk} \tag{7}$$

- Total energy consumed by a module for a given software is provided by Equation 8, where α_{module} is the total number of events in the software that access this module and τ_{module} is the task latency of this module. This equation is basically calculating the total number of cycles in the total execution time of the software for which this module is kept busy and multiplying that by the energy consumption of this module per clock cycle.

$$E_{module_{total}} = E_{module_{perClk}} \times \alpha_{module} \times \tau_{module} \tag{8}$$

- Total energy consumption of the entire system is the summation of the energy consumption of all the individual subsystems.

The above assumptions and equations are used for creating the energy models for individual subsystems, whose energy consumption measurements were made independent of other modules, using sense resistors that were placed on the power line connected only to that module. Some modules have significant idle or very-low-utilization energy consumption. For these modules the suitable correction factors are added in the equations.

The average energy consumption of the CORE subsystem including the Cortex-M3 cores, SGX540 graphics accelerators, image subsystem and face-detect subsystem was found to be 0.086J over one measurement sampling period, when this subsystem was idle. And the average increase in the energy consumption for a utilization of

around 92% was found to be 0.0257J over one measurement sampling period. So when these values are applied to the above equations we end up with the total energy prediction equation 11 for the CORE subsystem, where $\tau_{TotalTime}$ is the total time in seconds for which the software energy estimation is done, $\tau_{COREclk}$ is 2.6ns and N_{CORE} is the total number of CORE clock cycles ($f_{CORE_{clk}} \approx$ 384MHz) for which this subsystem is active.

$$E_{CORE_{Idle}} = (0.5158 \times \tau_{TotalTime})\, Joules \tag{9}$$

$$E_{CORE_{Active}} = (N_{CORE} \times 0.1683 \times \tau_{COREclk})\, Joules \tag{10}$$

$$E_{CORE} = (E_{CORE_{Idle}} + E_{CORE_{Active}})\, Joules \tag{11}$$

On similar lines, the LPDDR2 energy prediction equation 15 was generated by using the information that the idle/sleep average energy consumption of the LPDDR2 DRAM was measured to be 0.0191J over one measurement sampling period, the increase in the average energy when there was minimum activity with the LPDDR2 DRAM was found to be 0.0327J and the increase in average energy consumption from the minimum activity state energy consumption, when there was a utilization of 84% was found to be 0.0166J. $\tau_{SystemActiveTime}$ is the number of seconds for which the system was not in sleep. N_{LPDDR2} is equal to the number of L2 cache misses, when only the Cortex-A9 cores are accessing it. If the Cortex-M3 cores and the SGX540 graphics accelerators are being used, then these accesses need to be added to the number of L2 cache misses of the Cortex-A9 cores. Basically this is the number of memory requests to DDR2, which can be obtained by using any hardware performance counter libraries like OProfile. τ_{LPDDR2} is the average DRAM latency in DRAM clock cycles and $\tau_{LPDDR2clk}$ is 2.5ns as the DRAM clock is 400MHz.

$$E_{LPDDR2_{Idle}} = (0.1146 \times \tau_{TotalTime})\, Joules \tag{12}$$

$$E_{LPDDR2_{MinActivity}} = (0.1962 \times \tau_{SystemActiveTime})\, Joules \tag{13}$$

$$E_{LPDDR2_{Active}} = (N_{LPDDR2} \times \tau_{LPDDR2} \times 0.1169 \times \tau_{LPDDR2clk})\, Joules \tag{14}$$

$$E_{LPDDR2} = \left(E_{LPDDR2_{Idle}} + E_{LPDDR2_{MinActivity}} + E_{LPDDR2_{Active}}\right) Joules \tag{15}$$

The IVA subsystem was found to consume an average of 0.0064J of energy over one measurement sampling period. As we are not using the camera subsystem or LCD subsystem or audio subsystems along with the Pandaboard ES, IVA subsystem is of least interest and is not stressed by any microbenchmarks. But still the energy consumption of this subsystem is quite significant and should be considered in order to reduce the average error in the total energy prediction of the system. Equation 16 provides the energy estimation for the IVA subsystem. Here $\tau_{TotalTime}$ is the total time for which the software energy estimation is done.

$$E_{IVA} = (0.0384 \times \tau_{TotalTime})\, Joules \tag{16}$$

The Wi-Fi module was noticed to be consuming an average energy of 0.2005J over an interval of one measurement sampling period, when the system is in idle/sleep state. The increase in the average energy consumption, when the system was not in the sleep state and the Wi-Fi module was just maintaining a connection, was found to be 0.09J and under an utilization factor of 97% and at a transmission power of 20dbm, the average energy consumption still raised by around 0.9742J. It was noted earlier that the total energy consumption of the Wi-Fi module increases by an approximate

factor of 1.5, 2.5 and 4, when the transmission power is reduced to 10dbm, 5dbm and 0dbm respectively. We do not need to add any factor in the model to scale the total energy consumption for different transmit power settings, as the total energy consumption increases when the transmit power is reduced because of the increase in the total transmit time. $\tau_{WiFiTXTime}$ is the total number of seconds for which the Wi-Fi module transmitted/received data at a given transmit power. According to J.Huang et al. [23], the 3G wireless consumes around 5.3 times the power of the Wi-Fi and the 4G-LTE wireless consumes around 9.7 times the power of the Wi-Fi. These scaling factors can be provided as $\beta_{WiFiTo3G4Gscaling}$ to estimate the energy consumption of 3G or 4G LTE modules based on the energy consumption of the Wi-Fi module.

$$E_{WiFi_{Idle}} = (1.2027 \times \tau_{TotalTime})\, Joules \tag{17}$$

$$E_{WiFi_{MinActivity}} = (0.5399 \times \tau_{SystemActiveTime})\, Joules \tag{18}$$

$$E_{WiFi_{Active}} = (0.9742 \times \tau_{WiFiTXTime})\, Joules \tag{19}$$

$$E_{WiFi} = \left(\left(E_{WiFi_{Idle}} + E_{WiFi_{MinActivity}} + E_{WiFi_{Active}}\right) \times \beta_{WiFiTo3G4Gscaling}\right) Joules \tag{20}$$

The process of creation of energy models for the above modules was quite simple, as we had uncorrelated power measurement values of all these individual modules. But when it comes to the MPU subsystem, it consists of an instruction fetch unit, instruction decode unit, integer ALUs, MACs (multiply accumulate units), floating point units/NEON cores, load/store units, branch prediction units, 32KB L1 instruction and data caches, 1MB L2 cache and PL310 L2 cache controller. All these

submodules consume varied amount of energy when used, and since we do not have power measurements of these individual blocks inside the MPU subsystem, the energy prediction model for this subsystem cannot be created like done for other modules. Instead a linear regression energy model was created based on the energy consumption values obtained while stressing the integer ALU, the floation point/NEON unit, the L1 cache and the L2 cache separately with operations on varied datasets. The energy model generated for different sets of uncorrelated operations are shown below. Here N_{IA}, N_{IM}, N_{FAs}, N_{FAd}, N_{FMs}, N_{FMd}, N_{FDs}, N_{FDd}, N_{L1} and N_{L2} represent the total number of integer addition, integer multiplication, single precision floating point addition, double precision floating point addition, single precision floating point multiplication, double precision floating point multiplication, single precision floating point divide, double precision floating point divide, L1 accesses and L2 accesses performed by the software. ARM cortex-a9 does not have an integer divide instruction and thus the division will be done using software modules instead of using hardware. However, to obtain a rough energy estimate of the integer division instruction, one can consider them same as floating point divide, even-though this will cause over prediction of energy consumption. These numbers to be passed as parameters to the model, can be obtained using the OProfile events. The τ_{IA}, τ_{IM}, τ_{FAs}, τ_{FAd}, τ_{FMs}, τ_{FMd}, τ_{FDs}, τ_{FDd}, τ_{L1} and τ_{L2} are the instruction latencies of the integer addition, integer multiplication, single precision floating point addition, double precision floating point addition, single precision floating point multiplication, double precision floating point multiplication, single precision floating point divide, double precision floating point divide, L1 access and L2 access and are 1, 2, 4, 4, 5, 6, 15, 25, 4 and 19 respectively. This model does not take into account the instructions like register move. Such instructions can be treated as integer addition for energy prediction using this model. And the τ_{MPUclk} is the average clock period of the MPU subsystem under the current workload. If the MPU subsystem is working with a clock of 1.2GHz, then

τ_{MPUclk} will be 0.83ns.

$$e_{IA} = (N_{IA} \times \tau_{IA} \times 3.5857)\, Joules \tag{21}$$

$$e_{IM} = (N_{IM} \times \tau_{IM} \times 2.9693)\, Joules \tag{22}$$

$$e_{FA} = (((N_{FAs} \times \tau_{FAs}) + (N_{FAd} \times \tau_{FAd})) \times 3.1943)\, Joules \tag{23}$$

$$e_{FM} = (((N_{FMs} \times \tau_{FMs}) + (N_{FMd} \times \tau_{FMd})) \times 3.1493)\, Joules \tag{24}$$

$$e_{FD} = (((N_{FDs} \times \tau_{FDs}) + (N_{FDd} \times \tau_{FDd})) \times 2.7443)\, Joules \tag{25}$$

$$e_{L1} = (N_{L1} \times \tau_{L1} \times 4.7914)\, Joules \tag{26}$$

$$e_{L2} = (N_{L2} \times \tau_{L2} \times 3.8490)\, Joules \tag{27}$$

$$e_{Cortex-A9_{Idle}} = (0.0719 \times \tau_{TotalTime})\, Joules \tag{28}$$

$$e_{Cortex-A9_{Active}} = ((e_{IA} + e_{IM} + e_{FA} + e_{FM} + e_{FD} + e_{L1} + e_{L2}) \times \tau_{MPUclk})\, Joules \tag{29}$$

$$E_{MPU} = (e_{Cortex-A9_{Idle}} + e_{Cortex-A9_{Active}})\, Joules \tag{30}$$

The energy consumption of the rest of the Pandaboard ES system can be calculated using Equation 31.

$$E_{Rest} = (0.0724 \times \tau_{TotalTime})\, Joules \tag{31}$$

The total energy prediction of the entire system, for a given software, is just the sum of energy predictions for all the modules of the system.

$$E_{Total} = E_{MPU} + E_{CORE} + E_{LPDDR2} + E_{IVA} + E_{WiFi} + E_{Rest} Joules \tag{32}$$

The total power consumption of the system can be obtained by using the Equation 33.

$$\text{Total power consumed} = \frac{E_{Total}}{\tau_{TotalTime}} \tag{33}$$

The empirical values used in the model were tweaked by running different microbenchmarks and by trying to reduce the error of prediction of the model. Table 4 summarizes the parameters to be passed to the model and the method of obtain these parameters for a given software.

Table 4: Model parameters

Symbol	Description	Method of obtaining the parameter
$\tau_{TotalTime}$	Total amount of time for which energy estimation is to be done for a given software	User defined
$\tau_{SystemActiveTime}$	Total amount of time for which the system was not in sleep mode	Calculated using a system timer when the software is running
N_{CORE}	Number of CORE clock cycles for which the CORE subsystem is active	Total amount of time for which the CORE subsystem is active × 384MHz
N_{LPDDR2}	Number of memory requests to LPDDR2 DRAM	Performance monitoring events of ARM Cortex-A9 MPCORE
τ_{LPDDR2}	Average DRAM access latency	$\frac{((\text{Number of read requests to DRAM}\times 6)+(\text{Number of write requests to DRAM}\times 3))}{N_{LPDDR2}}$
$\tau_{WiFiTXTime}$	Total time for which the Wi-Fi transmits data	$\frac{\text{Total amount of bytes transferred using Wi-Fi module}}{\text{Bitrate of the Wi-Fi module}}$
$\beta_{WiFiTo3G4Gscaling}$	Wi-Fi to 3G or 4G-LTE energy scaling factor	1 if Wi-Fi is being used, 5.3 for 3G and 9.7 for 4G-LTE
N_{IA}	Number of integer additions (Register move operations can also be counted under this)	Performance monitoring events of ARM Cortex-A9 MPCORE
τ_{IA}	Integer addition latency	1
N_{IM}	Number of integer multiplications	Performance monitoring events of ARM Cortex-A9 MPCORE
τ_{IM}	Integer multiplication latency	2
N_{FAs}	Number of single precision floating point additions	Performance monitoring events of ARM Cortex-A9 MPCORE
τ_{FAs}	Single precision floating point addition latency	4
N_{FAd}	Number of double precision floating point additions	Performance monitoring events of ARM Cortex-A9 MPCORE
τ_{FAd}	Double precision floating point addition latency	4
N_{FMs}	Number of single precision floating point multiplications	Performance monitoring events of ARM Cortex-A9 MPCORE
τ_{FMs}	Single precision floating point multiplication latency	5
N_{FMd}	Number of double precision floating point multiplications	Performance monitoring events of ARM Cortex-A9 MPCORE
τ_{FMd}	Double precision floating point multiplication latency	6
N_{FDs}	Number of single precision floating point divisions	Performance monitoring events of ARM Cortex-A9 MPCORE
τ_{FDs}	Single precision floating point division latency	15
N_{FDd}	Number of double precision floating point divisions	Performance monitoring events of ARM Cortex-A9 MPCORE
τ_{FDd}	Double precision floating point division latency	25
N_{L1}	Number of L1 cache accesses	Performance monitoring events of ARM Cortex-A9 MPCORE
τ_{L1}	Typical L1 cache access latency	4
N_{L2}	Number of L2 cache accesses	Performance monitoring events of ARM Cortex-A9 MPCORE
τ_{L2}	Typical L2 cache latency	19
τ_{MPUclk}	Average MPU subsystem clock period	cpu_info_cur_freq system file of Android

The performance events to be monitored in the ARM Cortex-A9 MPCORE includes Main execution unit instructions (0x70), Second execution unit instructions (0x71), Floating-point instructions (0x73), NEON instructions (0x74) and memory related events (0x03, 0x04, 0x05, 0x06 and 0x07).

3.6 Accuracy of the Created Empirical Model

Several applications were created with a random mixture of different CPU instructions, graphics jobs and Wi-Fi download and upload tasks. These applications were run on android and the energy measurements of the different subsystems were taken. Then these measurements were compared with the energy prediction of the empirical model. The following graphs show the actual energy consumption and the energy prediction of the model for different subsystems.

BM1 benchmark runs 100 million each of integer additions, integer multiplications, single precision FP additions, multiplications and divisions. It also does 100 million accesses to random memory locations, out of which 18,749,348 requests were found to hit in L1, 33,418,973 hit in L2 and 47,831,679 requests were passed to the DRAM. After running these instructions, this benchmark downloads a 32.76MB file stored in an on-line server over Wi-Fi. The amount of time required by this benchmark to download this file was found to be 29.735s. The total time taken by the benchmark to run was 51.164s.

BM2 benchmark runs 100 million each of double precision FP additions, multiplications and divisions, single precision FP multiplications and integer additions. It also does 100 million accesses to random memory locations, out of which 12,375,924 requests were found to hit in L1, 38,164,448 hit in L2 and 49,459,628 requests were passed to the DRAM. After running these instructions, this benchmark downloads a 32.76MB file stored in an on-line server over Wi-Fi. The amount of time required by this benchmark to download this file was found to be 25.491s. It then sends this file over a socket connection over Wi-Fi. The time it took to completely send the file was found to be 52.337s at 20dbm transmit power. The total time taken by the benchmark to run was 99.139s.

BM3 benchmark runs 100 million each of single precision integer multiplications, double precision FP additions, single precision FP multiplications. It also does 100

million accesses to random memory locations, out of which 16,547,054 requests were found to hit in L1, 43,761,437 hit in L2 and 39,691,509 requests were passed to the DRAM. After running these instructions, this benchmark sent a file of size 53.41MB over a socket connection over Wi-Fi. The time it took to completely send the file was found to be 96.572s. The total time taken by the benchmark to run was 130.496s.

BM4 benchmark runs 100 million each of all the computational instructions. It also does 100 million accesses to random memory locations, out of which 16,988,458 requests were found to hit in L1, 40,931,650 hit in L2 and 42,079,892 requests were passed to the DRAM. After running these instructions, the benchmark downloads a 512MB file stored in an on-line server over Wi-Fi. The amount of time required by the benchmark to download this file was found to be 476.617s. It then sends this file over a socket connection over Wi-Fi. The time it took to completely send the file was found to be 963.434s. The total time taken by the benchmark was 1473.159s.

BM5 benchmark is similar to BM1, but instead of 100 million executions of each type of instruction, 200 million executions were performed. 28,674,024 memory requests were found to hit in L1, 76,375,195 requests were found to hit in L2 and the remaining 94,950,781 requests were passed to the DRAM. This benchmark took 28.982s to download the 32.76MB file and it took a total time of 77.125s to complete.

BM6 benchmark is similar to BM2, but instead of 100 million executions of each type of instruction, 200 million executions were performed. 25,072,961 memory requests were found to hit in L1, 72,137,299 requests were found to hit in L2 and the remaining 102,789,740 requests were passed to the DRAM. This benchmark took 24.416s to download the 32.76MB file and it took 55.879s to transmit this file over Wi-Fi at 20dbm of transmit power and a total time of 124.429s to complete.

BM7 benchmark is similar to BM3, but instead of 100 million executions of each type of instruction, 200 million executions were performed. 26,935,382 memory requests were found to hit in L1, 70,838,637 requests were found to hit in L2 and the

remaining 102,225,981 requests were passed to the DRAM. This benchmark took 231.752s to transmit a 53.41MB file over Wi-Fi at 5dbm of transmit power and a total time of 305.143s to complete.

BM8 benchmark is similar to BM4, but instead of 100 million executions of each type of instruction, 200 million executions were performed. 25,177,863 memory requests were found to hit in L1, 65,943,733 requests were found to hit in L2 and the remaining 108,878,404 requests were passed to the DRAM. This benchmark took 483.422s to download a 512MB file and it took 1509.838s to send this file over a socket connection via Wi-Fi at 10dbm of transmit power and a total time of 2063.961s to complete.

Figures 22 to 29 show the predicted and measured energy consumptions of different subsystems on Pandaboard ES for the benchmarks BM1 to BM8 respectively. The average energy prediction error was found to be 3.9418%, 8.2935%, -2.4082%, -7.3968%, -2.2060%, 17.4755% and -2.7457% for the MPU subsystem, CORE subsystem, LPDDR2 subsystem, Wi-Fi subsystem, IVA subsystem, rest of the Pandaboard ES system and the total Pandaboard ES system respectively.

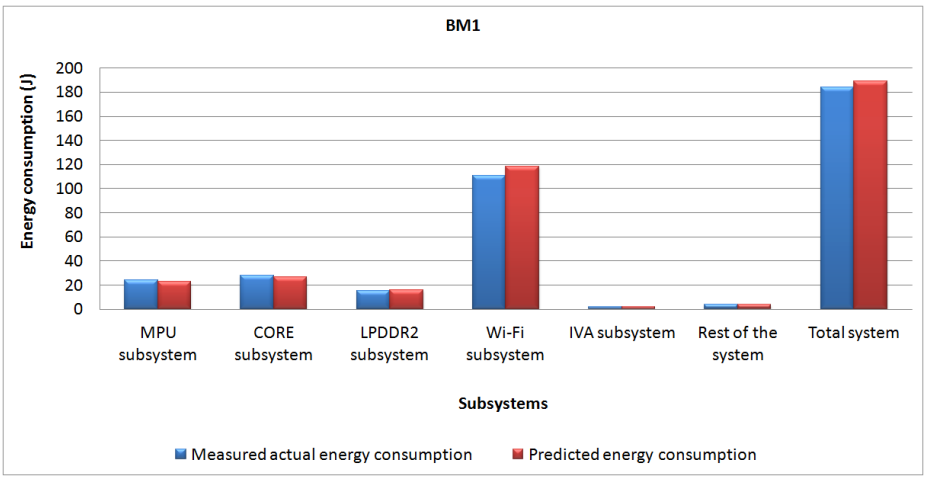

Figure 22: Actual measured and the model predicted energy consumption of different subsystems for the benchmark BM1

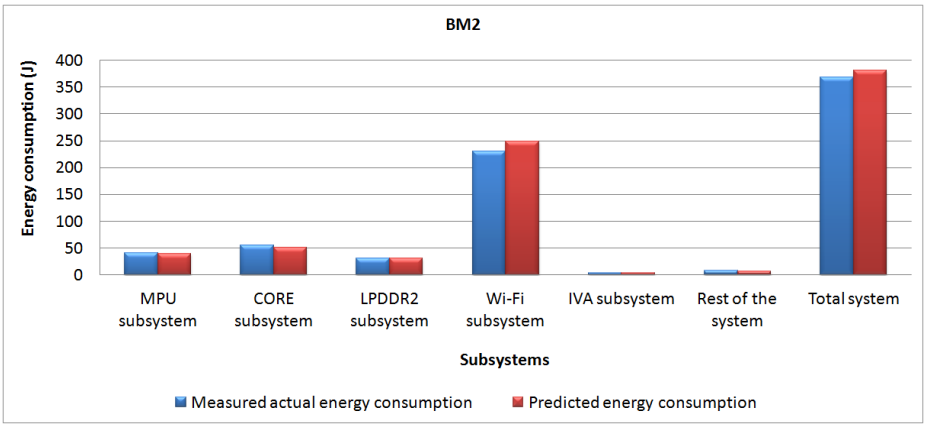

Figure 23: Actual measured and the model predicted energy consumption of different subsystems for the benchmark BM2

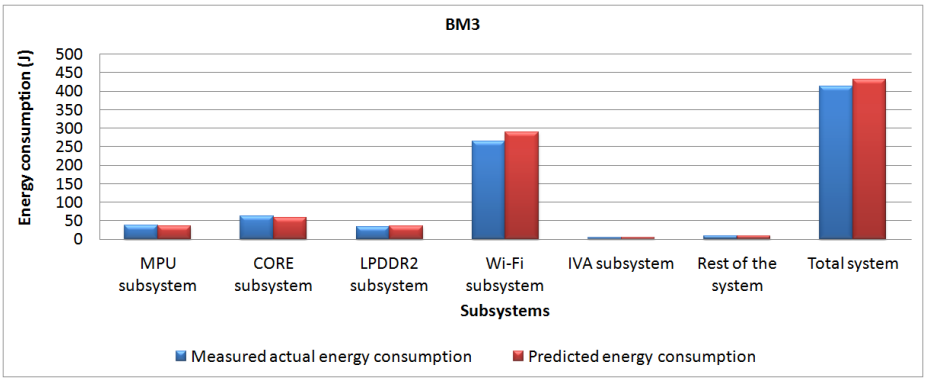

Figure 24: Actual measured and the model predicted energy consumption of different subsystems for the benchmark BM3

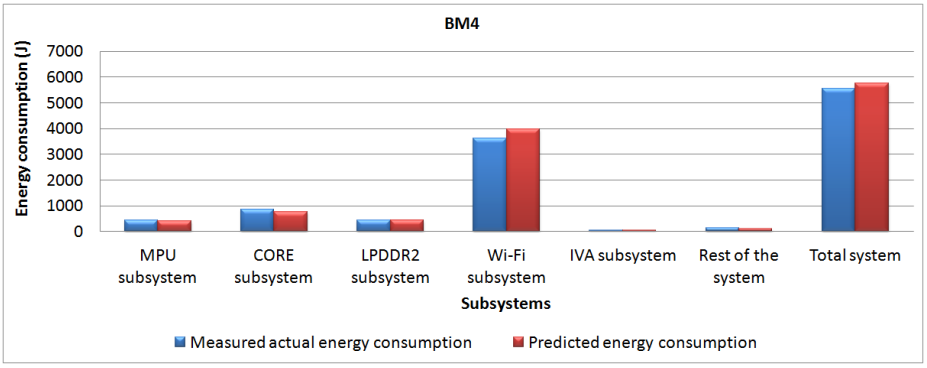

Figure 25: Actual measured and the model predicted energy consumption of different subsystems for the benchmark BM4

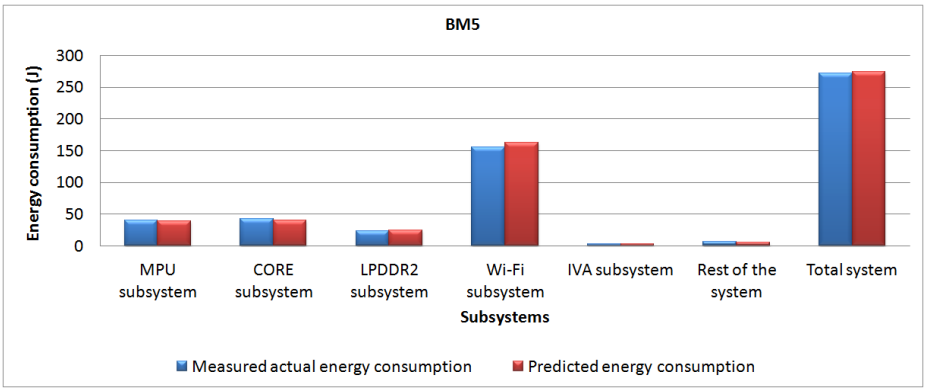

Figure 26: Actual measured and the model predicted energy consumption of different subsystems for the benchmark BM5

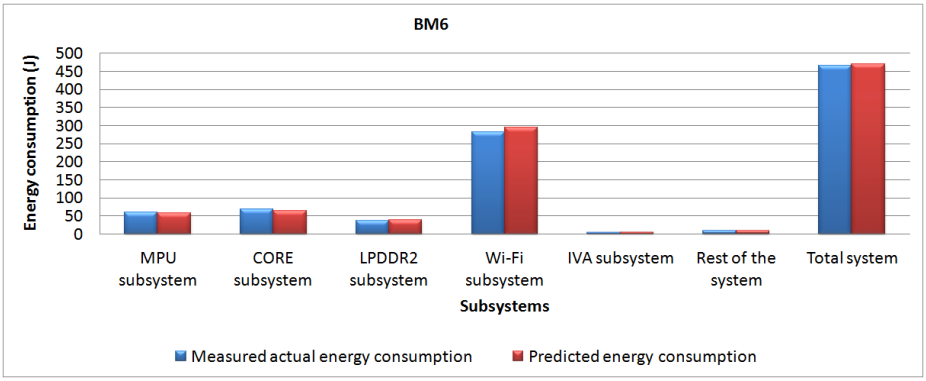

Figure 27: Actual measured and the model predicted energy consumption of different subsystems for the benchmark BM6

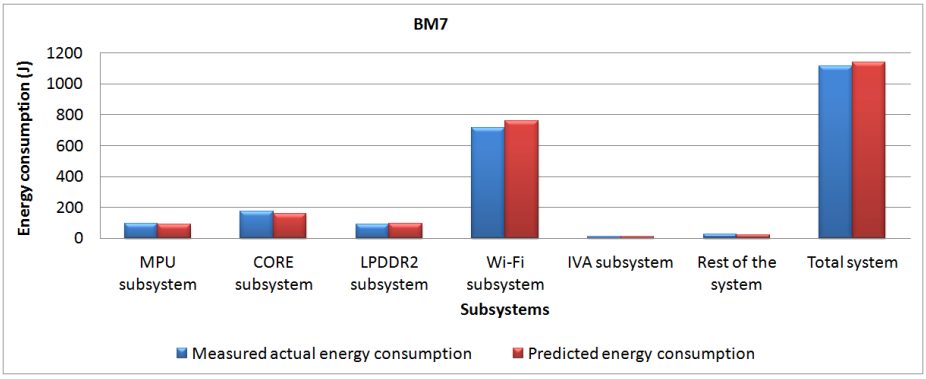

Figure 28: Actual measured and the model predicted energy consumption of different subsystems for the benchmark BM7

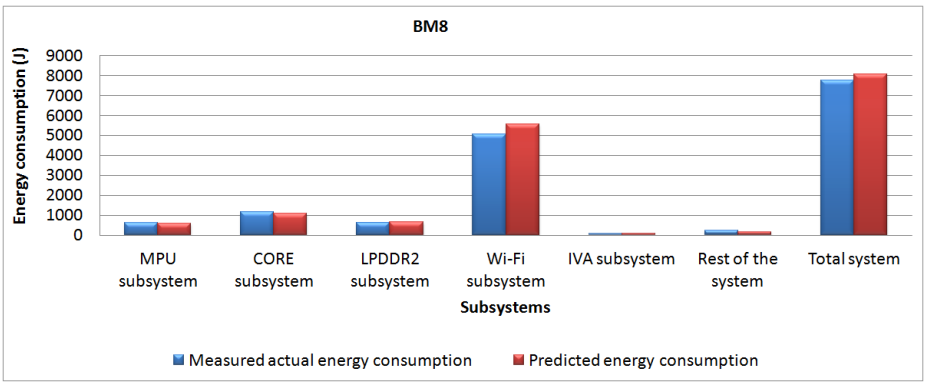

Figure 29: Actual measured and the model predicted energy consumption of different subsystems for the benchmark BM8

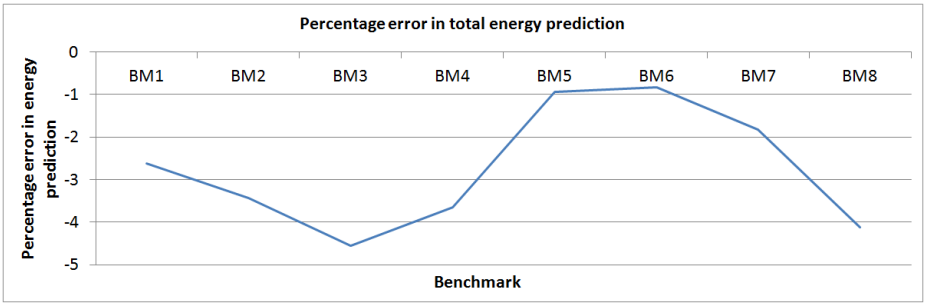

Figure 30: Percentage error in prediction of the total energy consumption of the Pandaboard ES system for variety of benchmarks

3.7 Limitations of the Model

The limitations of the model includes the fact that this model is biased towards the Pandaboard ES architecture and is expected to have higher error in prediction of energy consumption on different architectures. If this model is to be fine tuned for a different architecture, or a different IC being used for a subsystem, then all the experiments must be repeated for that subsystem. This model does not include an energy estimation model for LCD, camera or GPS subsystems. Thus the total energy estimation of the system, obtained from the model will not include energy data for LCD, camera and GPS modules. Hence this model will be useful to obtain the energy

estimation of individual subsystems and comparing the trend of energy consumption between them.

CHAPTER IV

RELATED WORK

Early works on power models concentrated on hardware at transistor, gate or RTL level design [16, 28, 29, 36]. The power estimation of these models are based on the circuit simulations and probabilistic analysis of signals [36]. Then came architectural or instruction level power models which are based on hardware power measurements of the systems and correlating the power consumption with the tasks performed on the system [15, 18–22, 24–27, 30–35, 37]. Software power estimation models arrived last with providing detailed power/energy consumption and distribution for different algorithms on a given architecture.

The pioneering work on software power estimation was done by Tiwari et al. [35]. This work created a power cost model for software for Intel 486DX2 and SPARClite934. Continuing this work, Tiwari et al. [34] generated an instruction level power model for a commercial 32-bit embedded microcontroller. Russel et al. [14] proposed a similar software energy estimation model created using hardware power measurements. This model clearly shows the effect a good compiler can have with regard to power consumption. Flinn et al. [22] created a profiling energy tool for applications called *PowerScope* using hardware power measurements correlated with statistically sampled system activity with kernel support. A similar approach was followed by F. Bellosa et al. [15] for creating a tool called *Joule Watcher* which uses the energy estimates to make intelligent scheduling of threads in-order to reduce energy consumption. Mortonosi et al. [25] proposed a run-time power estimation tool for high-performance microprocessors using the same approach.

More recent works on hardware/software power models, especially for mobile platforms include the work of Carroll et al. [18] which uses Openmoko Neo freerunner mobile phone to measure the power consumption of different subsystems and created an empirical power model for it. Zhang et al. [37] created a power model called *PowerBooter* based on sampling the battery voltage and calculating the energy remaining in it. They also created an Android application that works on this power model, to provide energy estimates of other applications running on the system.

Most of these work simply rely on the power measurements of the total system and creates empirical model based on this data. But we propose to measure the power consumption of individual modules on a mobile platform, similar to [18], so that we can achieve higher accuracy in power estimations.

CHAPTER V

CONCLUSION AND FUTURE RESEARCH

5.1 Conclusion

This work makes a comparative study of different methods of power/energy measurements and then proposes a power/energy measurement procedure in Chapter II that can be used in the process of creating a high accuracy power/energy estimation model. Chapter III provides the energy consumption data of different subsystems of Pandaboard ES obtained using the method explained in Chapter II. This chapter, then explains the steps of creating an empirical power/energy model out of the collected energy consumption values of different subsystems. The empirical model equations are produced in this chapter and also it shares the results of using the model on some of the custom benchmarks and calculates the average error in energy estimation for different subsystems of Pandaboard ES.

The average energy prediction error of the created empirical power/energy model was found to be 3.9%, 8.3%, -2.4%, -7.4%, -2.2%, 17.5% and -2.7% for the MPU subsystem, CORE subsystem, LPDDR2 subsystem, Wi-Fi subsystem, IVA subsystem, rest of the Pandaboard ES system and the total Pandaboard ES system respectively.

5.2 Future Research Directions

This thesis concludes with potential future research discussions.

5.2.1 Adding More Subsystems to the Empirical Model

Currently, the empirical power/energy model can provide energy estimates of the Cortex-A9 MPCORE processor subsystem, SGX 540 graphics subsystem, LPDDR2 memory subsystem and the Wi-Fi subsystem with high accuracy. However, it has

very basic energy estimation capabilities for the image, video and audio subsystem and the SD card subsystem. This model does not have energy estimation models for the LCD and speakers and thus makes estimates of just the image, video and audio back end subsystem without the LCD and speakers. Also the SD card energy estimation uses a simple average energy estimation model and does not consider the SD card usage statistics into account. The empirical models for these modules can be created and added to this model.

This model does not have energy estimation models for the GPS, 3G or 4G modules. However it just uses a energy consumption comparison factor between the Wi-Fi module and the 3G/4G-LTE modules as an estimate for the 3G/4G-LTE modules. The energy estimation models can be created for the GPS, 3G or 4G modules as discussed in Chapter 2II and Chapter 3III and added to the current model.

5.2.2 Finding or Using More Suitable Profilers

The energy/power model requires a lot of parameters regarding the software under investigation, obtained by code profilers, to provide an energy estimate of the software. These parameters includes the total time of execution, the total time of Wi-Fi transfers - which is in turn calculated by using the information of total number of bytes of data transmitted and received and the bit rates of these transactions, the profiled information on computational instructions and also memory profiled information. Currently, the software under investigation needs to be run with many profilers, and their logs need to be parsed to obtain the required information. However, it would be better to find or create a single profiler that can provide all the required information.

5.2.3 Integration with Android Emulator

This work had a vision of integrating the empirical power/energy model with the Android emulator as a library. In this case, the model will obtain all its parameters

directly from the emulator and provides its outputs with a GUI. This way any person, can run an Android application on the emulator and obtain the power/energy estimates of different subsystems for the given application on Pandaboard ES and need not worry about profiling the app and generating the model parameters based on the profile logs.

REFERENCES

[1] "0xbench - Integrated Android Benchmark Suite by 0xlab." https://code.google.com/p/0xbench/.

[2] "Aim I-prober 520 positional current probe." http://www.tti-test.com/go/iprober/.

[3] "Android Benchmark Suite." https://code.google.com/p/android-benchmark-suite/.

[4] "Cortex-A9 MPCORE Technical Reference Manual." http://infocenter.arm.com/help/topic/com.arm.doc.ddi0407g/DDI0407G_cortex_a9_mpcore_r3p0_trm.pdf.

[5] "EZ-probe positioner from Cascade Microtech." http://www.home.agilent.com/en/pd-1000004291%3Aepsg%3Apro-pn-E2654A/cascade-microtech-ez-probe-positioner?&cc=US&lc=eng.

[6] "HAL effect sensor." http:/en.wikipedia.org/wiki/Hall_effect_sensor.

[7] "Module test station from Cascade Microtech." http://www.x-web.nu/mttab_se/index.php?prodkate=698&prodid=5515.

[8] "NanoHttpd server library." https://github.com/NanoHttpd/nanohttpd.

[9] "OMAP4460 Technical Reference Manual." http://www.ti.com/pdfs/wtbu/OMAP4430_4460_4470_PUBLIC_TRM_Addendum_ABE_HAL_vF.pdf.

[10] "Pandaboard ES Reference Manual." http://pandaboard.org/sites/default/files/board_reference/pandaboard-es-b/panda-es-b-manual.pdf.

[11] "Split-core current probe TCP202." http://www.tek.com/datasheet/current-probe-dc-coupled-current-probe.

[12] "Tektronix DMM4040 6-1/2 digit precision multimeter." http://www.tek.com/digital-multimeter/dmm4050-4040/dmm4050-manual/dmm4040-and-dmm4050.

[13] "TiWi-R2-BLE Wi-Fi module." http://www.lsr.com/wireless-products/tiwi-ble.

[14] "Software power estimation and optimization for high performance, 32-bit embedded processors," in *Proceedings of the International Conference on Computer Design*, ICCD '98, (Washington, DC, USA), pp. 328–, IEEE Computer Society, 1998.

[15] Bellosa, F., "The benefits of event: driven energy accounting in power-sensitive systems," in *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, EW 9, (New York, NY, USA), pp. 37–42, ACM, 2000.

[16] Benini, L., Bruni, D., Chinosi, M., Silvano, C., Zaccaria, V., and Zafalon, R., "A power modeling and estimation framework for vliw-based embedded systems," in *ST Journal of System Research*, p. 118, 2001.

[17] Brooks, D., Tiwari, V., and Martonosi, M., "Wattch: a framework for architectural-level power analysis and optimizations," in *ISCA*, pp. 83–94, 2000.

[18] Carroll, A. and Heiser, G., "An analysis of power consumption in a smartphone," in *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, USENIXATC'10, (Berkeley, CA, USA), pp. 21–21, USENIX Association, 2010.

[19] Chang, N., Kim, K., and Lee, H. G., "Cycle-accurate energy measurement and characterization with a case study of the arm7tdmi," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 10, pp. 146–154, Apr. 2002.

[20] Contreras, G. and Martonosi, M., "Power prediction for intel xscale reg; processors using performance monitoring unit events," in *Low Power Electronics and Design, 2005. ISLPED '05. Proceedings of the 2005 International Symposium on*, pp. 221–226, 2005.

[21] Dong, M. and Zhong, L., "Sesame: Self-constructive system energy modeling for battery-powered mobile systems," *CoRR*, vol. abs/1012.2831, 2010.

[22] Flinn, J. and Satyanarayanan, M., "Powerscope: A tool for profiling the energy usage of mobile applications," in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, WMCSA '99, (Washington, DC, USA), pp. 2–, IEEE Computer Society, 1999.

[23] Huang, J., Qian, F., Gerber, A., Mao, Z. M., Sen, S., and Spatscheck, O., "A close examination of performance and power characteristics of 4g lte networks," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, MobiSys '12, (New York, NY, USA), pp. 225–238, ACM, 2012.

[24] Isci, C. and Martonosi, M., "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 36, (Washington, DC, USA), pp. 93–, IEEE Computer Society, 2003.

[25] Joseph, R. and Martonosi, M., "Run-time power estimation in high performance microprocessors," in *Proceedings of the 2001 international symposium on Low power electronics and design*, ISLPED '01, (New York, NY, USA), pp. 135–140, ACM, 2001.

[26] L.-B. Chen, Y.-L. C. and Huang, I.-J., "A real-time power analysis platform for power-aware embedded system development," *Journal of Information science and engineering*, vol. 1, no. 27, pp. 1165–1182, 2011.

[27] Lee, S., Ermedahl, A., Min, S. L., and Chang, N., "An accurate instruction-level energy consumption model for embedded risc processors," *SIGPLAN Not.*, vol. 36, pp. 1–10, Aug. 2001.

[28] Liu, D. and Svensson, C., "Power consumption estimation in cmos vlsi chips," *Solid-State Circuits, IEEE Journal of*, vol. 29, no. 6, pp. 663–670, 1994.

[29] Najm, F. N., "A survey of power estimation techniques in vlsi circuits," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 2, pp. 446–455, Dec. 1994.

[30] Pathak, A., Hu, Y. C., Zhang, M., Bahl, P., and Wang, Y.-M., "Fine-grained power modeling for smartphones using system call tracing," in *Proceedings of the sixth conference on Computer systems*, EuroSys '11, (New York, NY, USA), pp. 153–168, ACM, 2011.

[31] Rice, A. and Hay, S., "Decomposing power measurements for mobile devices," in *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, pp. 70–78, 2010.

[32] Simunic, T., Benini, L., and De Micheli, G., "Energy-efficient design of battery-powered embedded systems," in *Proceedings of the 1999 international symposium on Low power electronics and design*, ISLPED '99, (New York, NY, USA), pp. 212–217, ACM, 1999.

[33] Sinha, A. and Chandrakasan, A. P., "Jouletrack: a web based tool for software energy profiling," in *Proceedings of the 38th annual Design Automation Conference*, DAC '01, (New York, NY, USA), pp. 220–225, ACM, 2001.

[34] Tiwari, V. and Lee, M. T.-C., "Power analysis of a 32-bit embedded microcontroller," *VLSI Design Journal*, vol. 7, no. 3, pp. 225–242, 1998.

[35] Tiwari, V., Malik, S., and Wolfe, A., "Power analysis of embedded software: A first step towards software power minimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 437–445, 1994.

[36] Wang, H.-S., Peh, L.-S., and Malik, S., "A power model for routers: Modeling alpha 21364 and infiniband routers," *IEEE Micro*, vol. 23, pp. 26–35, Jan. 2003.

[37] Zhang, L., Tiwana, B., Qian, Z., Wang, Z., Dick, R. P., Mao, Z. M., and Yang, L., "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, CODES/ISSS '10, (New York, NY, USA), pp. 105–114, ACM, 2010.